
SERIALIZING VALUE OBJECTS IN RAILS

ARA HACOPIAN

@ahacop
SmartLogic

FAT MODELS

“[An] argument against Active Record is the fact that it couples the object design to the database design. This makes it more difficult to refactor either design...”

“[An] argument against Active Record is the fact that it couples the object design to the database design. This makes it more difficult to refactor either design...”

-- Martin Fowler, "Patterns of Enterprise Application Architecture", inventor of the Active Record pattern

SINGLE RESPONSIBILITY PRINCIPLE

SINGLE RESPONSIBILITY PRINCIPLE

There should never be more than one reason for a class to change

SINGLE RESPONSIBILITY PRINCIPLE

Not: All the methods that do something with a person go into the Person model

```ruby
class Person < ActiveRecord::Base
  def initials
    first_name.chr + middle_name.chr + last_name.chr
  end

  def armenian?
    if last_name.end_with?('ian')
      'probably'
    else
      'probably not'
    end
  end

  def weight_in_pounds
    weight
  end

  def weight_in_kilograms
    (weight * 0.453592).round
  end

  def weight_in_stone
    (weight * 0.07142).round
  end
end
```

COHESION

"WHERE SHOULD THIS METHOD GO?"

```ruby
class Person < ActiveRecord::Base
  def initials
    first_name.chr + middle_name.chr + last_name.chr
  end

  def armenian?
    if last_name.end_with?('ian')
      'probably'
    else
      'probably not'
    end
  end

  def weight_in_pounds
    weight
  end

  def weight_in_kilograms
    (weight * 0.453592).round
  end

  def weight_in_stone
    (weight * 0.07142).round
  end
end
```

HAS_ONE?

```
class Person < ActiveRecord::Base
  has_one :weight
end

class Weight < ActiveRecord::Base
  belongs_to :person
end
```

VALUE OBJECTS

» A small object that represents a simple value whose equality is based on its values rather than its identity

» Immutable

» Examples: addresses, money, names.

IDENTITY EQUALITY IN RUBY

```
> a = Object.new
> b = Object.new

> a.object_id
=> 70288883508240
> b.object_id
=> 70288892808780

> a == b
=> false
```

IDENTITY EQUALITY IN ACTIVERECORD

```
> george_foreman = Person.create
> george_foreman.id
=> 1

> ara = Person.create
> ara.id
=> 2

> ara == george_foreman
=> false
```

IDENTITY EQUALITY IN ACTIVERECORD

```
> george_foreman2 = Person.find(george_foreman.id)
> george_foreman2.id
=> 1

> george_foreman2 == george_foreman
=> true
```

IDENTITY EQUALITY IN ACTIVERECORD

```
> george_foreman.object_id
=> 70288888679260

> george_foreman2.object_id
=> 70288855436880

> george_foreman.object_id == george_foreman2.object_id
=> false
```

IDENTITY EQUALITY IN ACTIVERECORD

```
> george_foreman_jr = george_foreman.dup
> george_foreman_jr.id
=> nil

> george_foreman_jr == george_foreman
=> false
```

IDENTITY EQUALITY IN ACTIVERECORD

```
> george_foreman_iii = george_foreman.clone
> george_foreman_iii.id
=> 1
> george_foreman_iii == george_foreman
=> true
```

IDENTITY EQUALITY IN ACTIVERECORD

```
def ==(comparison_object)
  super ||
    comparison_object.instance_of?(self.class) &&
    !id.nil? &&
    comparison_object.id == id
  end
  alias :eql? :==
```

EXTRACT WEIGHT FROM PERSON

```ruby
class Weight
  attr_reader :pounds

  def initialize(pounds)
    @pounds = pounds
  end

  def kilograms
    (pounds * 0.453592).round
  end

  def stone
    (pounds * 0.07142).round
  end
end
```

WEIGHT EQUALITY

```
> buck_fifty = Weight.new(150)
> buck_fifty.pounds
=> 150
> buck_fifty.object_id
=> 70288888679260

> buck_fifty2 = Weight.new(150)
> buck_fifty2.pounds
=> 150
> buck_fifty2.object_id
=> 70288855436880
```

WEIGHT EQUALITY

```
> buck_fifty.pounds == buck_fifty2.pounds
=> true

> buck_fifty.object_id == buck_fifty2.object_id
=> false

> buck_fifty == buck_fifty2
=> false
```

WEIGHT EQUALITY

```
> weights = [Weight.new(150), Weight.new(150), Weight.new(150)]
> weights.uniq.length
=> 3

> pounds_values = [150, 150, 150]
> pounds_values.uniq.length
=> 1
```

WEIGHT AS A VALUE OBJECT

```ruby
class Weight
  include Comparable

  def <=>(other)
    other.instance_of?(self.class) && pounds <=> other.pounds
  end

  def eql?(other)
    self == other
  end

  def hash
    @hash ||= pounds.hash
  end
end
```

VALUE EQUALITY FOR WEIGHT

```
> buck_fifty = Weight.new(150)
> buck_fifty2 = Weight.new(150)

> buck_fifty.object_id == buck_fifty2.object_id
=> false

> buck_fifty == buck_fifty2
=> true
```

VALUE EQUALITY FOR WEIGHT

```
> weights = [Weight.new(150), Weight.new(150), Weight.new(150)]
> weights.uniq.length
=> 1
```

NAME AS A VALUE OBJECT

```ruby
class Name
  attr_reader :title, :first, :middle, :last, :suffix

  def initialize(title, first, middle, last, suffix)
    @title, @first, @middle, @last, @suffix = title, first, middle, last, suffix
  end

  def initials
    first.chr + middle.chr + last.chr
  end

  def armenian?
    if last.end_with?('ian')
      'probably'
    else
      'probably not'
    end
  end
```

NAME AS A VALUE OBJECT

```ruby
  def ==(other)
    other.instance_of?(self.class) &&
      title == other.title &&
      first == other.first &&
      middle == other.middle &&
      last == other.last &&
      suffix == other.suffix
  end
  alias :eql? :==

  def hash
    @hash ||= title.hash ^ first.hash ^ middle.hash ^ last.hash ^ suffix.hash
  end
end
```

3 WAYS TO SERIALIZE VALUE OBJECTS

1. Serialize
2. Virtual Attributes
3. Composed_of

SERIALIZE

```ruby
class Person < ActiveRecord::Base
  serialize :weight, Weight
end
```

```ruby
class Weight
  class << self
    def dump(weight)
      weight.pounds
    end

    def load(pounds)
      new(pounds)
    end
  end

  def initialize(pounds)
    @pounds = pounds
  end
```

```ruby
  attr_reader :pounds

  def kilograms
    (pounds * 0.453592).round
  end

  def stone
    (pounds * 0.07142).round
  end
end
```

```ruby
class Weight
  include Comparable

  attr_reader :pounds

  class << self
    def dump(weight)
      weight.pounds
    end

    def load(pounds)
      new(pounds)
    end
  end

  def initialize(pounds)
    @pounds = pounds
  end

  def kilograms
    (pounds * 0.453592).round
  end

  def stone
    (pounds * 0.07142).round
  end

  def <=>(other)
    other.instance_of?(self.class) && pounds <=> other.pounds
  end

  def eql?(other)
    self == other
  end

  def hash
    pounds.hash
  end
end
```

USING THE VALUE OBJECT WITH ACTIVERECORD

```
weight = Weight.new(150)
person = Person.create(weight: weight)
```

VIRTUAL ATTRIBUTES

```ruby
class Weight
  include Comparable

  attr_reader :pounds

  def initialize(pounds)
    @pounds = pounds
  end

  def kilograms
    (pounds * 0.453592).round
  end

  def stone
    (pounds * 0.07142).round
  end

  def <=>(other)
    other.instance_of?(self.class) && pounds <=> other.pounds
  end

  def eql?(other)
    self == other
  end

  def hash
    pounds.hash
  end
end
```

```
Person < ActiveRecord::Base
  def weight
    @weight ||= Weight.new(weight_value)
  end

  def weight=(other_weight)
    self.weight_value = other_weight.pounds

    @weight = other_weight
  end
end
```

```ruby
class Person < ActiveRecord::Base
  def name
    @name ||= Name.new(title, first_name, middle_name, last_name,
suffix)
  end

  def name=(other_name)
    self.title = other_name.title
    self.first_name = other_name.first
    self.middle_name = other_name.middle
    self.last_name = other_name.last
    self.suffix = other_name.suffix

    @name = other_name
  end
end
```

COMPOSED_OF

```ruby
class Person < ActiveRecord::Base
  composed_of :name,
    allow_nil: true,
    mapping: [
      %w(title title),
      %w(first_name first),
      %w(middle_name middle),
      %w(last_name last),
      %w(suffix suffix)
    ]

  composed_of :weight,
    allow_nil: true,
    mapping: %w(weight_value pounds)
end
```

```ruby
class Name
  attr_reader :title, :first, :middle, :last, :suffix

  def initialize(title, first, middle, last, suffix)
    @title, @first, @middle, @last, @suffix = title, first, middle, last, suffix
  end

  # other stuff...
end
```

QUESTIONS?